Remarks

Applicant respectfully requests reconsideration of this application as amended.

Claims 3, 15, and 19 have been amended. No claims have been cancelled or added.

Therefore, claims 1, 3-7, 13, 15,-17, 19-21, and 27-39 are presented for examination.

Claim Objections

Claim 3 stands objected to because it recites "The method of claim 2", and claim

2 was canceled by Applicant's amendment. Claim 15 stands objected to because it

recites "The system of claim 14", and claim 14 was canceled by Applicant's amendment.

Claim 19 stands objected to because it recites "The machine-readable medium of claim

18", and claim 18 was canceled by Applicant's amendment.

Claims 3, 15, and 19 have been amended to appear in proper form for allowance.

Therefore, applicant respectfully requests the objections to these claims be withdrawn.

35 U.S.C. §103(a) Rejection

Claims 1, 3-7, 13, 15-17, 19-21 and 27-39 stand rejected under 35 U.S.C. §103(a)

as being unpatentable over Archambault (U.S. Patent No. 6,173,444) in view of Blainey

(U.S. Patent No. 6,045,585) in view of "Restricted Pointers are Coming" by Robison

[hereinafter "Robison"]. Applicant submits that the present claims are patentable over

Archambault in view of Blainey and further in view of Robison.

Applicant submits that Archambault does not disclose or suggest using a *base*

pointer to determine whether an alias exists with a restricted pointer, as disclosed in

claim 1. In the Final Office Action, the Examiner states: "[p]ointer variables are

represented as l-values or r-values in the alias sets, and therefore the l-values and r-values

may be considered 'base pointers' for the pointer variables. In particular, the l-value of a pointer variable is a 'base pointer' because it is the address to which the variable points, or in other words, it is the value of the pointer." (Final Office Action mailed 6/24/2005 at page 3.)

However, as disclosed in the specification of the present application, "each pointer that is assigned the value of *another pointer*…in an instruction within a given code segment is said to have the *other pointer* as its 'base' pointer." (Specification at paragraph [0037].) The Examiner equates l-values and r-values in the alias sets of Archambault with the "base pointers" of the present application. Yet, the l-value of a pointer variable, or in other words, the address of the pointer variable, is not the same as a pointer that is assigned as a value to another pointer. The base pointers of the present application are other pointers assigned as a value, not an address of an actual pointer.

Furthermore, applicant submits that Archambault does not disclose or suggest *using scope* to determine whether an alias exists with a restricted pointer, as disclosed by claim 1. The Examiner states that "during intraprocedural analysis, Archambault determines aliases of pointer variables from the local scope of that procedure. During interprocedural analysis, Archambault determines aliases of pointer variables from more than one procedure, which is to say aliases of pointers from more than one scope." (Final Office Action at pg. 3.) In the preceding quote, the Examiner equates the "procedure" of Archambault with the "scope" of the present application. However, Archambault recites building a pointer alias graph for *each function* in a given set of code during the intraprocedural pass. (Archambault at col. 5, ll.4-6.) For example, Archambault recites, "an initial compilation phase in which intraprocedural information about pointer variables

referenced in *each function* of the program is gathered and saved in a data structure called the pointer alias graph." (Id. at col. 3 ln. 66-col. 4 ln. 2.)

Therefore, during the intraprocedural analysis of Archambault, alias analysis of pointers only occurs for each function in a given listing of code. As the term "function" was not separately defined within Archambault, Applicant submits that the ordinary meaning of the term to one skilled in the art should be applied. Generally, a "function" within software refers to the sequence of code that performs a particular task, as part of a larger program, and is grouped into one or more statement blocks. The National Institute of Standards and Technology defines a "function" as "a computation which takes some arguments or inputs and yields an output," and further as "a subroutine which returns a value."

However, as disclosed in the specification of the present application, "[s]cope is the portion of the program to which a declaration applies. In an embodiment, in the C programming language, scope is denoted by an open brace ('{') followed by a closed brace ('}'). There may be many open and close braces nested together denoting inner scope and outer scopes." (Specification at paragraph [0033].) Comparing this use of "scope" with the ordinary meaning of a "function", it is clear that the two terms do not refer to the same thing. Significantly, a function may actually include many inner and outer scopes.

Consequently, Archambault is not using scope to determine whether an alias exists with a restricted pointer. In fact, as referred to in the background of the present application, a particular problem of compiling restricted pointers using prior art compilation techniques is that inner and outer scopes are not considered. (Specification

at paragraph [0003].) The method of Archambault does not disclose a method that addresses this particular problem, as it does not perform alias analysis based on <u>scopes</u> within a particular section of code. Instead, it performs alias analysis of <u>functions</u> within a listing of code.

Therefore, Archambault does not disclose or suggest the features of claim 1. Nor do Blainey or Robison disclose or suggest the features of claim 1, such as using a *base pointer* to determine whether an alias exists with a restricted pointer and *using scope* to determine whether an alias exists with a restricted pointer. As a result, Archambault, Blainey, and Robison, individually or in combination, do not disclose or suggest the features of claim 1. Therefore, claim 1 is patentable over Archambault and Blainey, in view of Robison. Claims 3-7 and 27-31 depend from claim 1 and include additional limitations. As such, claims 3-7 and 27-31 are also patentable over Archambault and Blainey, in view of Robison.

Independent claims 13 and 17 contain similar features to claim 1, such as using a *base pointer* to determine whether an alias exists with a restricted pointer and *using scope* to determine whether an alias exists with a restricted pointer. As discussed above, Archambault, Blainey, and Robison, individually or in combination, do not disclose or suggest such features. Therefore, claims 13 and 17 are also patentable over Archambault and Blainey, in view of Robison, for the reasons discussed with respect to claim 1.

Claims 15, 16, and 37-39 depend from claim 13 and include additional limitations. Claims 19-21 and 32-36 depend from claim 17 and include additional limitations. Therefore, claims 15, 16, 19-21, and 32-39, are also patentable over Archambault and Blainey, in view of Robison.

Applicant respectfully submits that the rejections have been overcome and that the claims are in condition for allowance. Accordingly, applicant respectfully requests the rejections be withdrawn and the claims be allowed.

The Examiner is requested to call the undersigned at (303) 740-1980 if there remains any issue with allowance of the case.
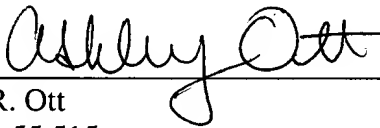
Applicant respectfully petitions for an extension of time to respond to the outstanding Office Action pursuant to 37 C.F.R. § 1.136(a) should one be necessary. Please charge our Deposit Account No. 02-2666 to cover the necessary fee under 37 C.F.R. § 1.17(a) for such an extension.

Please charge any shortage to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: August 2, 2005

Ashley R. Ott
Reg. No. 55,515

12400 Wilshire Boulevard
7th Floor
Los Angeles, California 90025-1026
(303) 740-1980